Cika Electrónica

Nota de Aplicación: CAN-059
Título: **Utilización de displays LCD gráficos (HD61202) con Ramtron VRS51L3074**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	08/02/07	port de CAN-003

Nos interiorizaremos ahora en el desarrollo de una interfaz para conectar un módulo LCD gráfico inteligente a un microcontrolador Ramtron VRS51L3074. Utilizaremos un módulo Powertip PG12864, de 128x64 pixels, basado en chips controladores compatibles con el HD61202, de Hitachi, y su clon: el KS0108, de Samsung. Analizaremos más tarde el software de control y un simple programa demostración, que sirve para comprobar el correcto funcionamiento de los módulos LCD que tengamos en stock, y de paso, demostrar sus capacidades gráficas. A fin de probar la mayor parte posible del hardware, la interfaz será de 8 bits y realizará lectura y escritura del controlador LCD.

Hardware

Dado que el controlador utilizado en estos módulos sólo es capaz de direccionar 64x64 pixels, este tipo de displays utiliza dos controladores, uno para la parte izquierda y otro para la parte derecha del display. El hardware de conexión resulta casi igual al utilizado para los displays alfanuméricos, es decir, la clásica interfaz tipo 6800, pero con la introducción de tres nuevas señales: dos señales de selección del controlador (CS1 y CS2) y una de reset (RST). La señal RS de los displays alfanuméricos tiene el nombre I/D en los displays gráficos (Instruction/Data), pero su funcionamiento es el mismo.

Otra diferencia es el circuito de contraste; si bien los módulos alfanuméricos funcionan muy bien con una tensión fija cercana a los 0,6V, los módulos gráficos necesitan de una tensión de aproximadamente -8V. La misma puede obtenerse de la salida que estos módulos proveen (Vout ó Vee).

Para la interfaz con el micro, deberemos tener en cuenta el controlador que posee el display. Dado que el micro es de 3,3V y 5V-tolerant, podemos usar displays basados en KS0108 sin problemas, dado que éste tolera los niveles de tensión del VRS51L3074, pero si nuestro display posee un HD61202, deberemos intercalar algún traslador de nivel como por ejemplo 74HCT245, y controlar el sentido de habilitación de éste.

Ramtron	LCD
P0.0	D0
P0.1	D1
P0.2	D2
P0.3	D3
P0.4	D4
P0.5	D5
P0.6	D6
P0.7	D7
P2.0	I/D
P2.1	R/W
P2.2	E
P2.3	CS1
P2.4	CS2

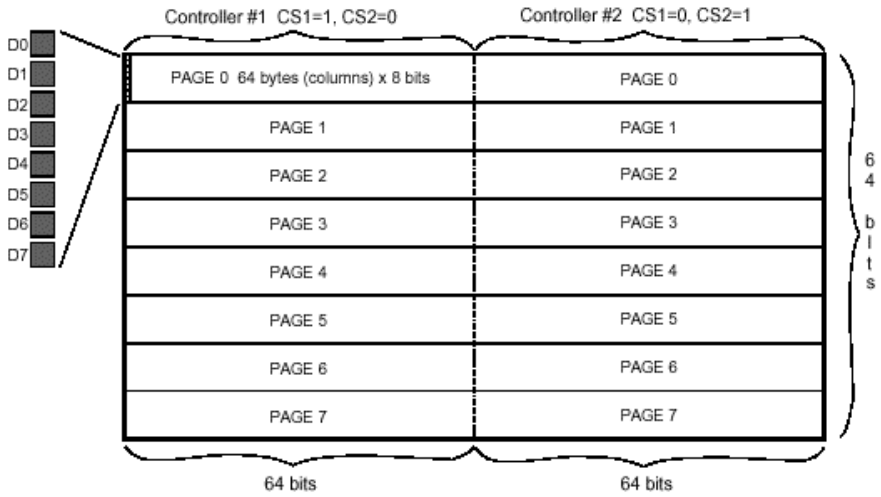
Software

Estructura y breve descripción del display gráfico

La estructura de memoria de estos módulos gráficos es algo caprichosa, la misma se halla agrupada en bytes en sentido vertical, divididos a su vez en páginas. Debido al hecho de que, a su vez, se necesitan dos controladores, la pantalla resulta dividida a la mitad, y cada mitad es atendida por un controlador.

El bit menos significativo del primer byte de memoria del primer controlador corresponde al punto situado en la pantalla arriba a la izquierda, y el bit más significativo del último byte de memoria del segundo controlador corresponde al punto situado en pantalla abajo a la derecha.

En el gráfico que figura a continuación puede apreciarse un esquema de esta estructura:



El direccionamiento del byte a leer o escribir en memoria se hace mediante comandos, como en los displays alfanuméricos, pero aquí tenemos dos direccionamientos: vertical y horizontal, y por lo tanto dos comandos :

1. setear dirección horizontal (de 0 a 63)
2. setear dirección vertical o número de página (de 0 a 7)

El direccionamiento horizontal tiene un contador autoincrementado, el cual apunta a la dirección siguiente luego de una lectura o escritura. Esto resulta óptimo para enviar los datos byte por byte hasta completar una página.

Algoritmos

Para direccionar un punto horizontalmente, basta con comparar su coordenada horizontal *x* con 64, y seleccionar uno u otro controlador. Una vez seleccionado el controlador correspondiente, la dirección horizontal dentro de ese controlador corresponde a la coordenada horizontal *x*, si *x*<64, o a *x*-64, si *x*>=64.

Para ubicar un punto verticalmente, deberemos dividir la coordenada vertical *y* por 8, siendo la parte entera del resultado el número de página y el resto el número de bit dentro del byte seleccionado.

Para graficar funciones, debemos tener en cuenta que la coordenada (0;0) se halla en el extremo superior izquierdo de la pantalla.

Para mostrar pantallas, deberemos agrupar los datos de modo tal de poder enviarlos de forma que aproveche de manera eficiente los contadores autoincrementados y la estructura de memoria:

- 64 bytes correspondientes a la primera página del controlador izquierdo (bit menos significativo corresponde a la parte superior)
- 64 bytes correspondientes a la primera página del controlador derecho (bit menos significativo corresponde a la parte superior)
- repetir para las 7 páginas siguientes, total de 1024 bytes.

Si bien sería más eficiente mandar primero la información a un controlador y luego al otro, resulta menos complicado convertir las pantallas de un formato común a este formato. El lector puede optar por la forma de implementación que más le agrade.

Para imprimir textos, podemos recurrir a un simple truco que nos permita simplificar el software: definimos líneas de texto o “renglones”, y nos limitamos a escribir dentro de ellas. Cada línea de texto corresponderá a una página de memoria del controlador, y si utilizamos una tipografía de 7 puntos de alto, la hacemos caber perfectamente dentro de los 8 bits de la página, manteniendo un punto de separación entre líneas. En sentido horizontal, haremos que un caracter tenga todos sus puntos de un mismo controlador. Si utilizamos una tipografía de 5x7, con un punto de separación entre caracteres, tendremos 6 puntos por caracter en sentido horizontal; dentro del espacio de un controlador (64 puntos), podremos dibujar 10 caracteres (60 puntos), y 4 pixels sin utilizar a ambos lados. Nuestro display gráfico puede entonces superponer textos en una matriz imaginaria de 20 caracteres por 8 líneas.

Para simplificar aún más, seteamos las direcciones al momento de escribir cada caracter; este esquema tal vez no sea el más eficiente, pero resulta extremadamente simple y es ideal para demostrar las capacidades de estos displays.

Desarrollo

Desarrollamos a continuación el software de base para manejo del display, elegimos el lenguaje C dado qu el micro es mucho más rápido que el display, aprovechando la posibilidad de poder intercalar assembler donde nos simplifica las cosas. Si el usuario así lo desea, puede utilizar assembler en todo el código.

```
/* LCD control signals */
#define LCD_CS1        3
#define LCD_CS2        4
#define LCD_E          2
#define LCD_ID         0
#define LCD_RW         1

/* LCD status bits */
#define BUSY   7


/* Low level functions */

void LCD_SelSide(unsigned char side)
{
       if(side)       {
              P2|=(1<<LCD_CS2);      // Sube CS2
              P2&=~(1<<LCD_CS1);     // Baja CS1
       }
       else   {
              P2|=(1<<LCD_CS1);      // Sube CS1
              P2&=~(1<<LCD_CS2);     // Baja CS2
       }
}

void LCD_CalcPage(unsigned char y,unsigned char *page,unsigned char *row)
{
       *page=(y>>3);                  // page = parte entera de y coord /8
       *row=y-((*page)<<3);           // row = resto
}

void LCD_Write(unsigned char dat)
{
  /* HD61202U: 150ns address setup, 200ns data setup, 450ns E width */

        P0=dat;                       // escribe data
       P0PINCFG=0;                    // P0 = Outputs, pone datos en el bus
       P2&=~(1<<LCD_RW);              // Baja RW (Write), ~50ns delay @40MHz
       _asm
         DA A                         ; 100ns delay, 4 ciclos @40Mhz
       _endasm;
       P2|=(1<<LCD_E);                // Sube E (50ns, total 200ns)
       _asm
         DA A
         DA A                         ; 400ns delay, 16 ciclos @40Mhz
          DA A
         DA A
       _endasm;
       P2&=~(1<<LCD_E);               // Baja E (50ns, total 450ns)
       _asm
         DA A
         DA A                         ; 400ns delay, 16 ciclos @40Mhz
         DA A                         ; RET agrega más demora
         DA A
       _endasm;
}


unsigned char LCD_Read()
{
  /* HD61202U: 150ns address setup, 450ns E width, 320ns data delay */

unsigned char dat;
```

```
        P2|=(1<<LCD_RW );               // Sube RW (Read)
        P0PINCFG=0xFF;                  // P0 = Inputs ~50ns delay @40MHz
        _asm
          da a                            ; 100ns delay, 4 ciclos @40Mhz
         _endasm;
        P2|=(1<<LCD_E);                 // Sube E (50ns, total 200ns)
        _asm
          DA A
          DA A                          ; 300ns delay, 12 ciclos @40Mhz
           DA A
         _endasm;
        dat=P0;                          // lee datos, ~75ns @40MHz
        _asm
          NOP                           ; 25ns delay, 1 ciclo @40Mhz processor
         _endasm;
        P2&=~(1<<LCD_E);                // Baja E (50ns, total 450ns)
        _asm
          DA A
          DA A                          ; 400ns delay, 16 ciclos @40Mhz
          DA A                          ; return() agrega más demora
          DA A
         _endasm;
        return(dat);
}


unsigned char LCD_Status()
{
unsigned char status;

         P2&=~(1<<LCD_ID);              // Baja ID (Cmd)
        do {
               status=LCD_Read();       // lee status (flag de busy)
        } while(status&(1<<BUSY));      // espera busy=0
        return(status);
}

unsigned char LCD_ReadData()
{
        LCD_Status();                   // vuelve con ID=0
        P2|=(1<<LCD_ID);                // Sube ID (Data)
        return(LCD_Read());
}

void LCD_WriteCmd(unsigned char cmd)
{
        LCD_Status();                   // vuelve con ID=0
        LCD_Write(cmd);
}

void LCD_WriteData(unsigned char dat)
{
        LCD_Status();                   // vuelve con ID=0
        P2|=(1<<LCD_ID);                // Sube ID (Data)
        LCD_Write(dat);
}
```

Luego las funciones de inicialización y de soporte de alto nivel.

La inicialización del módulo LCD se realiza por hardware, mediante el pin RST. El estado de reset puede comprobarse mediante los bits de status; no obstante, por simpleza, asumimos que el controlador se ha reseteado exitosamente.

```
void LCD_init ()
{
        P0=0;
         P2&=0xE0;                      // Salidas = 0
         P2PINCFG&=0xE0;                // xxx00000 P2.0,1,2,3,4 = output
        //MsDelay ( 1000 );             // demora para reset del LCD

        LCD_SelSide(1);
        LCD_WriteCmd (0x3F);            // Display on
        LCD_SelSide(0);
        LCD_WriteCmd (0x3F);            // Display on
```

```
}

void LCD_home ( void )
{
	LCD_SelSide(1);
	LCD_WriteCmd (0x40);		// address 0
	LCD_WriteCmd (0xC0);		// la pantalla comienza en address 0
	LCD_SelSide(0);
	LCD_WriteCmd (0x40);		// address 0
	LCD_WriteCmd (0xC0);		// la pantalla comienza en address 0
}

void LCD_fill(unsigned char pattern)
{
unsigned char i,page;
	LCD_home();					// address 0,0
	for(page=0;page<8;page++) {
		LCD_SelSide(0);			// controlador izquierdo
		LCD_WriteCmd (0xB8+page);	// page address
		for(i=0;i<64;i++)
			LCD_WriteData(pattern);	// pattern
		LCD_SelSide(1);			// controlador derecho
		LCD_WriteCmd (0xB8+page);	// page address
		for(i=0;i<64;i++)
			LCD_WriteData(pattern);	// pattern
	}
}

#define LCD_clear() LCD_fill(0)


void LCD_plot (unsigned char x, unsigned char y)
{
unsigned char page,row,dat;

	LCD_SelSide(x>63);			// 0-63 => side 0; 63>127 => side 1
	x=(x>63)?(x-64):x;			// corrige x si x >63
	LCD_WriteCmd (0x40+x);			// address = x o x-64
	LCD_CalcPage(y,&page,&row);		// calcula página y fila dentro de la página
	LCD_WriteCmd (0xB8+page);		// page address
	LCD_ReadData();				// dummy read luego de setear la dirección
	dat=LCD_ReadData();			// lee dato
	dat|=(1<<row);				// setea bit
	LCD_WriteCmd (0x40+x);			// vuelve a setear address (leer inc contador)
	LCD_WriteData(dat);
}

void LCD_dump (unsigned char * __code imgdata)
{
unsigned char i,page;

	LCD_home();					// address 0,0
	for(page=0;page<8;page++) {
		LCD_SelSide(0);			// controlador izquierdo
		LCD_WriteCmd (0xB8+page);	// page address
		for(i=0;i<64;i++)
			LCD_WriteData(*(imgdata++));	// mitad izquierda de la página
		LCD_SelSide(1);			// controlador derecho
		LCD_WriteCmd (0xB8+page);	// page address
		for(i=0;i<64;i++)
			LCD_WriteData(*(imgdata++));	// mitad derecha de la página
	}
}

void LCD_putchr ( char chr )
{
unsigned char i;
	for(i=0;i<5;i++)
		LCD_WriteData(font5x7s[5*(chr-0x20)+i]);	// escribe caracter
	LCD_WriteData(0);					// separador
}

void LCD_printat (unsigned char row, unsigned char col, char *ptr)
{
```

```
unsigned char x;
        do {
                x=(col>9)?6*(col-10):6*col;          // corrige address si col>9
                if(col>9)                            // 0-9 => side 0; 10>20 => side 1
                        LCD_SelSide(1);
                else {
                        LCD_SelSide(0);
                        x+=4;                        // dejo un borde de 4 pixels a los lados
                }
                LCD_WriteCmd (0xB8+row);             // page address
                LCD_WriteCmd (0x40+x);               // address = 6*col or 6*col-10
                LCD_putchr (*ptr++);
                col++;
        }       while (*ptr);
}
```

El siguiente fragmento de código es un simple ejemplo para escribir un corto y simple programa de control que nos permita corroborar el funcionamiento de un módulo LCD gráfico inteligente. Para observar si todos los pixels funcionan correctamente, pintamos la pantalla de negro y luego la blanqueamos. Luego graficamos una función coseno y finalmente mostramos una pantalla. Los bitmaps y fonts fueron generados según la información brindada en las CAN-022 y CAN-041, respectivamente.

```
/* MAIN PROGRAM */

main()
{
int i;
extern __code unsigned char Cika[],Ramtron[];


        LCD_init();

         while(1){

                LCD_clear();
                MsDelay ( 3000 );                    // espera 3 secs
                LCD_fill(0xFF);                      // pinta de negro
                MsDelay ( 3000 );                    // espera 3 secs
                LCD_clear();

                for(i=0;i<128;i++)
                        LCD_plot(i,32);              // eje x
                for(i=0;i<64;i++)
                        LCD_plot(64,i);              // eje y
                for(i=-64;i<63;i++)
                        LCD_plot(64+i,32-(int)(32*(cosf(i*PI/64))));
                                                     // plot cos(x); 0,0 arriba  la izquierda
                LCD_printat(0,0,"cos(x)");
                MsDelay ( 3000 );                    // espera 3 secs
                LCD_dump(Cika);
                MsDelay ( 3000 );                    // espera 3 secs
                LCD_dump(Ramtron);
                LCD_printat(7,2,"Cika Electronica");
                LCD_printat(0,0,"Ramtron Int Corp");
                MsDelay ( 3000 );                    // espera 3 secs
        }
}
```